AMENDMENTS TO THE CLAIMS

Claims 1 and 3-42 are pending in the Application. A complete listing of the current pending claims is provided below and supersedes all previous claim listing(s.) No new matter has been added.

1. (Currently Amended) A method for reducing IOs by coalescing writes in a computer system, comprising:

identifying, in a first storage location, a first data block to be written into a second storage location, the first data block having a first data block address and being associated with a first destination address;

identifying, in the first storage location, an additional data block to be written into the second storage location, the additional data block having a second data block address and being associated with a second destination address;

tracking a total number of the identified first and additional data blocks; and

writing the identified first and additional data blocks to the second storage location according to the first and second destination addresses with a write operation, in which said first and additional data addresses constitute ~~blocks comprising a set of data blocks with~~ consecutive ~~data block~~ addresses and the first and additional destination addresses constitute consecutive addresses.

2. (Cancelled)

3. (Previously Presented) The method of claim 1, further comprising setting a predetermined upper limit for the total number of the identified first and additional data blocks, in which if the predetermined upper limit is met by the total number of the identified first and additional data blocks, the method stops identifying additional data blocks and writes the identified first and additional data blocks to the second storage location.

4. (Original) The method of claim 1, further comprising copying each of the identified first and additional data blocks to a temporary storage location, wherein writing the identified first and additional data blocks to the second storage location is accomplished

by writing copies of the identified first and additional data blocks in the temporary storage location to the second storage location.

5. (Original) The method of claim 4, further comprising marking the identified first and additional data blocks in the first storage location not dirty each time after an identified data block is copied to the temporary storage location.

6. (Original) The method of claim 5 in which the temporary storage location is a temporary buffer cache.

7. (Original) The method of claim 1, wherein identifying additional data blocks comprises:

sequentially searching the first storage location to identify next data blocks with higher and lower data block addresses consecutive to the first data block address; and

determining whether an identified next data block is dirty,

wherein if the identified next data block is not dirty, the method stops sequentially searching for new data blocks with data block addresses at a same data block address side to the data block address of the identified next data block.

8. (Original) The method of claim 7 in which sequentially searching the first storage location is conducted alternatively at the lower and higher data block address sides to the first data block address.

9. (Original) The method of claim 8 in which sequentially searching the first storage location begins by searching the next data block at the lower data block address side.

10. (Original) The method of claim 8 in which sequentially searching the first storage location begins by searching the next data block at the higher data block address side.

11. (Original) The method of claim 7 in which sequentially searching the first storage location comprises:

searching for all dirty next data blocks with consecutive data block addresses to the first data block address at one of the lower and higher data block address sides; and

searching for all dirty next data blocks with consecutive data block addresses to the first data block address at another one of the lower and higher data block address sides.

12. (Original) The method of claim 1 in which the first storage location is a buffer cache.

13. (Original) The method of claim 1 in which the second storage location is a disk.

14. (Original) The method of claim 1 in which the computer system is a database system.

15. (Original) The method of claim 1 further comprising:

copying the first and additional data blocks to form a copy of the first and additional data blocks; and

allowing other entities in the computer system to access the copy of the first and additional data blocks during the act of writing the first and additional data blocks.

16. (Original) The method of claim 1 wherein identifying the additional data blocks comprises:

sequentially searching the first storage location to identify next data blocks in the direction of either the higher or lower data block addresses consecutive to the first data block address.

17. (Original) The method of claim 16 wherein if an identified next data block is not dirty, sequentially searching in the opposite direction of either the higher or lower data block addresses consecutive to the first data block address.

18. (Original) The method of claim 1 in which the computer system comprises a database system.

19. (Original) The method of claim 1 in which the first storage location is a buffer cache, wherein the entire buffer cache is searched.

20. (Original) The method of claim 19 in which hashing is performed to search the entire buffer cache.

21. (Currently Amended) A computer system, comprising:

means for identifying, in a first storage location, a first data block to be written into a second storage location, the first data block having a first data block address and being associated with a first destination address;

means for identifying, in the first storage location, an additional data block to be written into the second storage location, the additional data block having a second data block address and being associated with a second destination address;

means for tracking a total number of the identified first and additional data blocks; and

means for writing the identified first and additional data blocks to the second storage location according to the first and second destination addresses with a write operation, in which said first and additional data addresses constitute ~~blocks comprising a set of data blocks with~~ consecutive ~~data block~~ addresses and the first and additional destination addresses constitute consecutive addresses.

22. (Previously Presented) The computer system of claim 21, further comprising:

means for setting a predetermined upper limit of the total number of the identified first and additional data blocks, in which if the predetermined upper limit is met by the total number of the tracking means, the computer system stops identifying additional data blocks and immediately writes the identified first and additional data blocks to the second storage means.

23. (Previously Presented) The computer system of claim 21, further comprising:

temporary storage means for temporarily storing data in a third plurality of data blocks;

means for copying each of the identified first and additional data blocks to the temporary storage means; and

means for marking one of the identified first and additional data blocks in the first storage means not dirty after the one of the identified first and additional data blocks copied to the temporary storage means, in which the identified first and additional data blocks are written to the second storage means by copying the identified first and additional data blocks in the temporary storage means to the second storage means.

24. (Original) The computer system of claim 23, wherein the temporary storage means is a temporary buffer cache.

25. (Previously Presented) The computer system of claim 21, wherein said identifying means comprises:

means for sequentially searching the first storage means to identify the first data block and next data blocks with higher and lower data block addresses consecutive to the first data block address of the first data block; and

means for determining whether the an identified next data block is dirty, in which if the identified next data block is not dirty, the computer system stops sequentially searching for new data blocks with data block addresses on a same side of the first data block where the at least one next data block resides.

26. (Original) The computer system of claim 25, wherein the sequentially searching means searches the first storage means alternatively at the lower and higher data block address sides to the first data block address.

27. (Original) The computer system of claim 25, wherein the sequentially searching means searches the first storage means for all dirty data blocks with consecutive data block addresses to the first data block address at one of the lower and higher data block address side before it searches the first storage means for all dirty data blocks with consecutive data block addresses to the first data block address at another one of the lower and higher data block address sides.

28. (Original) The computer system of claim 21 in which the first storage means is a buffer cache.

29. (Original) The computer system of claim 21 in which the second storage means is a disk.

30. (Original) The computer system of claim 21 in which the computer system is a database system.

31. (Currently Amended) A computer program product comprising a ~~computer usable~~ computer-readable storage medium having executable code to execute a process for reducing IOs by coalescing writes in a computer system, the process comprising:

identifying, in a first storage location, a first data block to be written into a second storage location, the first data block having a first data block address and being associated with a first destination address;

identifying, in the first storage location, an additional data block to be written into the second storage location, the additional data block having a second data block address and being associated with a second destination address;

tracking a total number of the identified first and additional data blocks; and

writing the identified first and additional data blocks to the second storage location according to the first and second destination addresses with a write operation, in which said first and additional data addresses constitute ~~blocks comprising a set of data blocks with~~ consecutive ~~data block~~ addresses and the first and additional destination addresses constitute consecutive addresses.

32. (Previously Presented) The computer program product of claim 31, in which the process for reducing IOs further comprises:

setting a predetermined upper limit of the total number of the identified first and additional data blocks, in which if the predetermined upper limit is met by the total number of the tracking means, the computer system stops identifying additional data

blocks and writes the identified first and additional data blocks to the second storage location.

33. (Previously Presented) The computer program product of claim 31, in which the process for reducing IOs further comprises:

temporarily storing data in one or more third data blocks;

copying each of the identified first and additional data blocks to a temporary storage location; and

marking one of the identified first or additional data blocks in the first storage location not dirty after the one of the identified first or additional data block is copied to a temporary storage location, in which the identified first and additional data blocks are written to the second storage location by copying the identified first and additional data block in the temporary storage location to the second storage location.

34. (Previously Presented) The computer program product of claim 33 comprising a computer usable storage medium having a executable code to execute a process for reducing IOs, in which said one or more third data blocks are stored in a temporary buffer cache.

35. (Previously Presented) The computer program product of claim 31 comprising a computer usable storage medium having a executable code to execute a process for reducing IOs, in which the steps of identifying data blocks further comprising:

sequentially searching the first storage location to identify the first data block and at least one next data block with higher and lower data block address consecutive to the first data block address of the first data block; and

determining whether the an identified next data block is dirty, in which if the at least one next data block is not dirty, the computer system stops sequentially searching for new data blocks with data block addresses on a same side of the first data block where the at least one next data block resides.

36. (Previously Presented) The computer program product of claim 35, in which said step of sequentially searching the first storage searches the first storage location alternatively at the lower and higher data block address sides to the first data block address.

37. (Previously Presented) The computer program product of claim 35, in which said step of sequentially searching searches the first storage location for all dirty data blocks with data block addresses consecutive to the first data block address on one of the lower and higher data block address side before searching the first storage location for all dirty data blocks with data block addresses consecutive to the first data block address on another one of the lower and higher data block address sides.

38. (Previously Presented) The computer program product of claim 31, in which said first storage is a buffer cache.

39. (Previously Presented) The computer program product of claim 31, in which said second storage is a disk.

40. (Previously Presented) The computer program product of claim 31, in which said computer is a database system.

41. (New) The method of claim 1, in which the first and second data block addresses are logical addresses.

42. (New) The method of claim 1, in which the first and additional addresses are physical addresses.